# Software Interactions and the GNU General Public License

*Malcolm Bain*[a]

*(a) Partner, id law partnres, Barcelona.*

DOI: 10.5033/ifosslr.v2i2.44

**Abstract**
This article presents the current results of the work carried out within a working group of the European Legal Network of lawyers, facilitated by the Free Software Foundation Europe, whose aim is to provide some general guidance to lawyers and developers working with free software to understand the technical and (potentially) legal effects of the interaction or interoperation of two programs in the context of GPLv2 licensing.[1]

**Keywords**
GPL; linking; copyright, Free and Open Source Software

**Info**
This item is part of the **Articles** section of IFOSS L. Rev. For more information, please consult the relevant section policies statement. This article has been independently peer-reviewed.

The so-called "GPL linking" debate has been raging for the last 18 years, and probably will go on for a quite a few more. It has been seriously considered by legal authors such as, among others, Lawrence Rosen in "*Open Source Licensing: Software Freedom and Intellectual Property Law*"[2]

1 This article is based on the work carried out in the context of the Software Interactions working group of the Free Software Foundation Europe, of which I am "rapporteur", and takes from the resulting *Working Paper on the legal implications of certain forms of Software Interactions (a.k.a linking)*", which is available online at <http://www.ifosslr.org/public/LinkingDocument.odt>. I would like to thank those participating in this work group for their input and feedback, however all opinions and errors made herein are my own. Special thanks go to Neil Brown, Andrew Katz and Martin von Willebrand for their comments on this paper.

2 Rosen, Lawrence (2004), *'Open Source Licensing, Software Freedom and Intellectual Property Law'* , Prentice Hall, available online at <http://www.rosenlaw.com/oslbook.htm>

and "*The unreasonable fear of infection*",[3] or Dan Ravicher, now of the Software Freedom Law Center, in an LWT interview in 2003, "*Dan Ravicher on derived works*".[4] It has been hotly argued on discussion lists such as Debian-legal and in comments on *LWT, Groklaw or Slashdot*,[5] and (more politely?) on Open Source Initiative's license discussion list[6] and the FSF-Europe European Legal Network's own discussion list. In Europe, authors on the subject include Andrew Katz in an article for the Society for Computers and Law, "*GPL – the Linking Debate*",[7] and Mikko Välimäki in *'GNU General Public License and the Distribution of Derivative Works'.*[8]

The question at the heart of the matter is under what circumstances, if a software program or application "uses" GPL'd code (and I use the vague word "use" on purpose here, as I comment on this below), does this use cause the application to be covered by the copyleft provisions of the GPL – either as a derivative work or otherwise – and thus render any redistribution of the application subject to the GPL.

Art.2b of the GPLv2 provides:

> *2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:*
>
> …
>
> *b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.*
>
> …
>
> *These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program,*

3 Rosen, Lawrence (2001) 'The unreasonable fear of infection', *Linux Journal,* available online at <http://rosenlaw.com/html/GPL.PDF>

4 Available online at http://lwn.net/Articles/62202/. Other US based articles include Determann, Lothar (2006): *'Dangerous Liasons--Software Combinations as Derivative Works? Distribution, Installation, and Execution of Linked Programs Under Copyright Law, Commercial Licenses, and the GPL'* Berkeley Technology Law Journal, Volume 21, issue 4; online at <http://www.btlj.org/data/articles/21_04_03.pdf>

5 E.g.: on LWN: '*GPL and linking'* (Feb 16, 2006), at http://lwn.net/Articles/172226/: Slashdot: *'WordPress Creator GPL Says WP Template Must Be GPL'd'* (July 22, 2010), at <http://yro.slashdot.org/story/10/07/22/1935248/WordPress-Creator-GPL-Says-WP-Template-Must-Be-GPLd>

6 Archives available at <http://www.mail-archive.com/license-discuss@opensource.org/>

7 Katz, Andrew (2007) *'GPL - The Linking Debate'*, SCL Magazine, Vol 18 Issue 3; available online at http://www.scl.org/site.aspx?i=cl0

8 Välimäki, Mikko, '*GNU General Public License and the Distribution of Derivative Works'*, 2005 (1) The Journal of Information, Law and Technology (JILT), available online at <http://www2.warwick.ac.uk/fac/soc/law2/elj/jilt/2005_1/välimäki/>.

> *the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.*
>
> *Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.*
>
> *In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.*

Understanding this is a key question for any developer who wishes to "use" a GPL component in her own application, as this has an impact not just on the licensing of the resulting work, but also implications for license compliance such as providing source code and a copy of the license, and the resulting negative consequences (legal and/or reputational) for getting it wrong.

This article does not comment substantively on this debate, but mainly reports on the work carried out by the "Software Interactions Working Group" of the aforementioned Freedom Task Force to bring some light to the matter. The substantive work, which we will call for ease the "Software Interactions Document", focuses on the interpretation of GPLv2 and has been presented and published as a work-in-progress by the FSFE.[9]

## 1. Presenting the Software Interactions Document

### 1.1 Purpose and scope

The aim of the Software Interactions Document is to provide some general guidance to lawyers and developers working with free software to understand the technical and (potentially) legal effects of the interaction or interoperation of two programs. While there is a general awareness of the issue among serious users of free software, we have found there are a lot of misconceptions, both in the legal and IT engineering communities, regarding the scope, impact, effect and obligations surrounding the use of GPL'd software. The document aims to clear up some of these misconceptions and note consensus if and where there is consensus on any aspect, and highlight areas of debate that may usually be linked to the specifics of each case.

More specifically, the purpose of the work is to facilitate understanding of different mechanisms of interaction between programs in order to assist decision making, in free and non-free software projects, for intermediaries within the software supply chain and end-users, who use or intend to use GPL'd software programs, as to whether a program may or must be considered a derivative work of another (original) work, or possibly a collective (composite) work incorporating a

9 *'Working Paper on the legal implication of certain forms of Software Interactions (a.k.a linking)'*, available online at <http://www.ifosslr.org/public/LinkingDocument.odt>. The analysis is mainly based on the European legal framework established by Council Directive of 14 May on the legal protection of computer programs (91/250/EEC) Official Journal L 122 , 17/05/1991 p. 42-46 ("EUCPD", consolidated in Directive 2009/24/EC Official Journal L 111 , 5/5/2009 p. 16-22).

previous work, or whether it could be considered independent. Even more specifically, it aims to shed some light on the use of GPLv2'd software components, or creating software for GPLv2 platforms, and the scope of the copyleft provisions as established in this license.

The document is descriptive and exploratory, focussing on a limited number of interaction mechanisms,[10] and it does not aim to establish any legal or normative position or "doctrine" in the matter – it presents a step by step legal analysis of the combination of two software components and the considerations which could or would be taken into account, as we describe below. The actual legal effect of any form of interaction will depend on the circumstances of each case, and the work only provides preliminary (and simplified) examples of code.

In addition, the legal interpretation and consequences of any form of interaction (e.g. whether it creates a derivative work or not, under which license a program may be distributed, what are the distribution obligations) will depend on the specific legal framework of the jurisdiction (state) in which the question arises, whether during the course of developing new software or in copyright infringement proceedings. For example, certain jurisdictions may not grant copyright protection for certain aspects or elements of a work (e.g. in the USA, "*processes, systems and methods of operation*",[11] in the EU member states, "*ideas and principles… underlying [a program's] interfaces*"[12]) which may limit the scope of exclusive control of a copyright holder.

Finally, the document also includes comment on the so-called "community view"– i.e. the opinion of the members of the community from which the software is taken or in which the software is developed, taking into account that there is not necessarily a single representative community voice. Where possible, we have tried to indicate where there is a divergence. Due to the nature of the free software environment, from a business point of view the community view may be of equal if not more relevance than the strict legal interpretation of a license, for the purpose of assessing risks and benefits when taking a decision about the licensing and distribution of inter-related components of software.

### 1.2 Challenges

Work on the document has been no easy task, with a number of challenges.

First, we have found that there is no clear technical definition or consensus on certain (or any!) forms of software interactions, with many forms of implementation, exceptions, special cases, contexts, programming paradigms and languages.

Second, each of the persons participating in the work has brought a different legal background and tradition to the table, with a different approach to asking and answering questions towards resolving the issue, and a different legal vocabulary and set of case law. While copyright law has

10 The document currently looks at static and dynamic linking, Remote Procedure Calls, system calls, macro and template expansions, "plug-ins" and interpreted language communication mechanisms. As technology evolves and further input is provided, the aim is to expand the analysis to further interactions, if necessary.

11 Section 102 of the U.S. Copyright Act (title 17 of the U.S. Code). Relevant comment can be found in Samuelson, Pamela (2007) *'Why Copyright Law Excludes Systems and Processes from the Scope of its Protection',* Texas Law Review, Vol. 85, No. 1, available online at <http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1002666>.

12 Article 1.1, Council Directive 91/250/EEC of 14 May 1991 on the legal protection of computer programs.

been "harmonised" to a certain extent internationally,[13] it is not sufficiently so either to provide a single method of legal reasoning or juridical approach (for example, the US 9th Circuit Abstraction-Filtration-Comparison test[14] would not necessarily - or at all - be used by courts in Spain or France to determine copying or creation of derivative works) nor, even when using the same approach, to come to a single interpretation of the law to a theoretical series of facts.

Third: the "Community" view. Decisions relating to the use or non-use of GPL'd code, like most decisions, are not based purely on legal arguments, but also significantly on a risk analysis that takes into account the views of the free software community as a whole (which may eventually be consecrated as a "trade custom") and by the community of the specific GPL'd software that is to be used. There is a difference between dealing with a vociferous, if not necessarily legally correct, community (and possibly just a minority of members) and one where there is space to discuss and reach a consensus on the matter at hand.[15]

## 2. Substantive issue

The main issue addressed by the Software Interactions Document revolves around the following question: does a specific form of software interaction or interoperation create a work that, if and when distributed, must be so under the copyleft provisions of GPLv2 (and when does it not)?

This question has arisen for two main reasons: first, there is no clear-cut answer to what is a "derivative work", as defined by copyright law, or "work based on another" (and even if there were, this could vary according to jurisdiction) and second, GPLv2 itself is not clear (or rather, has multiple definitions) regarding what it considers falls within the copyleft obligations of redistribution of the whole under the terms of the GPLv2 (Art. 2b in particular).

The Free Software Foundation, drafter of GPLv2, gives its view on the issue in the GPL-FAQs.[16]

> *You have a GPL'd program that I'd like to link with my code to build a proprietary program. Does the fact that I link with your program mean I have to GPL my program?*
>
> *Yes.*

13 The Berne Convention (last amended 1979) and WIPO Copyright Treaty of 1996 – texts available online at <http://www.wipo.int/treaties/en/ip/berne/trtdocs_wo001.html> and <http://www.wipo.int/treaties/en/ip/wct/trtdocs_wo033.html> respectively. At European level, Council Directive 91/250/EEC of 14 May 1991 on the legal protection of computer programs, available online at <http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31991L0250:EN:HTML>, consolidated in Directive 2009/24/EC (<http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2009:111:0016:0022:EN:PDF>).

14 Discussed in Ravicher, Dan (2002) '*Software Derivative Work: A Jurisdiction Dependent Determination*', 1, (Nov. 2002), Linux.com, online at <http://www.linux.com/archive/feature/113252>. See also Omar Johnny, Marc Miller, Mark Webbink (2010) '*Copyright in Open Source Software - Understanding the Boundaries*', IFOSSLR Vol 2, N°1, available online at <http://www.ifosslr.org/ifosslr/article/view/30>, DOI: 10.5033/ifosslr.v2i1.30

15 Examples of community debate include: 'Linux: the GPL and binary modules' at <http://kerneltrap.org/node/1735> and NDIS Wrapper at <http://kerneltrap.org/Linux/NDISwrapper_and_the_GPL>.

16 FSF, GPLv2 FAQs: online at <http://www.gnu.org/licenses/old-licenses/gpl-2.0-faq.html>, section titled *'Combining work with code released under the GPL'*.

While this is a general answer, and the FAQs themselves enter into more discussion on the issue[17] the Software Interactions Document attempts to analyse a subset of these interactions according to the methodology set out below.

## 2.1. Five Steps

For the purpose of the Document, by way of methodology, the question is broken down into five main questions or steps, the first four looking at copyright law and the fifth looking at additional relevant wording of GPLv2.

1. *What is the original software artefact that is being used in the new work, is it protected by copyright, and to what extent?*
2. *When creating and/or distributing the new work including or interacting with the original software artefact, is any act restricted by copyright being performed in relation to that software artefact, and if so, which?*
3. *Still within the borders of copyright law, if there is no clear-cut answer to these questions, at what additional test or criteria might a court look to determine if the new work could be considered to be the result of the performance of an act restricted by copyright (reproduction or transformation)?*
4. *If you have established that the work in question is protected by copyright, and that the act which you are looking to perform is an act restricted by copyright then, irrespective of any purported grant of licence / permission, does the creation or use of the original work amount to fair use, fair dealing or is any other defense available in the relevant jurisdiction?*
5. *Having done the "bare" copyright-based analysis, set out in the preceding questions, we can finally ask what, if anything, does the wording of the GPL add to this copyright-based analysis (particularly if the answer is in the negative, or at least not clear), and how can that wording be interpreted?*

We look at these questions in turn below:

***1. What is the original software artefact that is being used in the new work, is it protected by copyright, and to what extent?***

This question raises several issues. The scope of copyright protection is jurisdiction specific. Generally speaking, under the international treaties, works in the public domain and "ideas and principles" underlying the software are not protected (including, under the EU Computer Programs Directive, those that "underlie its interfaces"[18]). In the second case (ideas and principles) the scope of these concepts is not clearly defined. US legislation, which excludes any "idea, procedure, process, system, method of operation, concept, principle, or discovery" as mentioned above,[19] and courts (and authors?) seem to have been more active in determining these boundaries, and have

17 Subsequent FAQs, from <http://www.gnu.org/licenses/old-licenses/gpl-2.0-faq.html#LinkingWithGPL> onwards, e.g.: *'What is the difference between "mere aggregation" and "combining two modules into one program*?'

18 EUCPD, Art. 1

19 US Copyright Act, 17 U.S.C., Section 102(b)

excluded (a) purely functional elements, (b) ideas (when merged with the expression) (c) "scenes a faire", (d) works in the public domain, and (e) facts, among other limits on copyright protection.[20]

2. ***When creating and/or distributing the new work including or interacting with the original software artefact, is any act restricted by copyright being performed in relation to that software artefact (i.e. is there a clear infringement: reproduction, transformation, distribution?)**, **and if so, which? (merely copying, or copying and transforming?)***

I.e. does creating and redistributing the (combined/inter-related) work involve the performance of an act restricted by copyright, *stricto sensu* (other than distribution of the software artefact itself), regardless of what the GPL may otherwise add. In particular we ask if a particular form of software interaction, under a pure or "bare" copyright law analysis, creates a derivative work of one or both of the interacting software components. This is because the license at least is clear that it applies to derivative works "under copyright law" (here read: strict interpretation of legislation/case law).

Regarding this question, again we find a difference between jurisdictions regarding the creation of a derivative work or "transformation". While the US law states that a derivative work is a "a work based upon one or more pre-existing works"[21] (giving rise to tests of substantial similarity and inclusion and a certain amount of interesting case law), the EUCPD talks of "the translation, adaptation, arrangement and any other alteration of a computer program and the reproduction of the results thereof".[22] Within the EU jurisdictions, it seems there is a distinct lack of case law on derivative works of computer programs.

***3. Still within the borders of copyright law, if there is no clear-cut answer to these questions, at what additional test or criteria might a court look to determine if the new work could be considered to be the result of the performance of an act restricted by copyright (reproduction or transformation)?***

This will be even more case specific. In English law, for example, this may be seen within the context of "non-verbatim copying" or similar tests. Here, we could mention, for example:

- *dependency/independency criteria* (does the new work function without the incorporated/inter-related GPL work? Could you swap the GPL component for another one? If so or if not, to what extent? Is there a non-protected API being used as part of the interaction?),
- "*critical functionalities*" (does the GPL component provide critical functionalities for the new work – are these functions more than mere "scenes a faire" or "methods", that might be excluded under applicable law?),
- "*made for*" (has the plug in been made for a GPL core/kernel, and if so, which part of the core? Does the design of the artefact for which the plug in has been made exert such an influence on the design and development of the plug in that the second developer is (ab)using the skill and judgement of the first?), or

20 Discussed in Samuelson, Pamela (2007) ibid.; and Omar, Johnny, et al. ibid.
21 US Copyright Act, 17 U.S.C., Section 101
22 EUCPD, Article 4

- "*use or reproduction of a substantial part of the skill, labour and judgment invested in the original work*" when developing the new work.

or other such rationale that (relevant) courts may have used in case law (e.g. copyright is also interested in the manner in which a work is created – which is why there are clean room developments - and not just with which artefact the work interacts or what it does once created, so it could look at the development process ).

Note that questions 2 and 3 overlap, or at least it is difficult and even artificial to separate the answers, certainly in case-based jurisdictions where court decisions also establish the law (as opposed to interpreting it).

***4. If you have established that the work in question is protected by copyright, and that the act which you are looking to perform is an act restricted by copyright then, irrespective of any purported grant of licence / permission, does the creation or use of the original work amount to fair use, fair dealing or is any other defense available in the relevant jurisdiction?***

Again, we meet several challenges as the exemptions from copyright infringement vary from jurisdiction to jurisdiction. In the US one would first look to rely on "fair use" or other explicit exemptions (in the UK "fair dealing" exemptions), while in other EU countries legislation tends to have created a series of specific exemption use-cases, most of which are not relevant for our purposes, but usually include exemptions in favour of interoperability. And there may also be a *de minimis* exception, whereby trivial reproduction will not be covered (in England/Wales, extended by exemption for "insubstantial copying").

***5. Having done the "bare" copyright-based analysis, set out in the preceding questions, we can finally ask what, if anything, does the wording of the GPL add to this copyright-based analysis (particularly if the answer is in the negative, or at least not clear), and how can that wording be interpreted?***

We know that with a GPL'd work, there will always be an infringement defence (authorisation) prior to distribution, as the license permits reproducing and transformation... however, an important question is: what are the conditions on exploitation of the third party GPL code (or the plug in for the GPL code)? This is because, for instance, the conditions on copying and distribution are different from those on modifying and distribution. To answer this, we would go back to both the answer to question 2 (which act restricted by copyright is performed?) and the wording of the GPL, and try to resolve any conflicting language.

This is a key question because the GPL purports to cover not only "works based on the Program" as interpreted by copyright law, but also works that "in whole or in part contain … the Program or any part thereof", leading us to look into the question of collective/composite works[23] (also possibly considered derivative works- not necessarily the result of an "adaptation/transformation", but because of the "inclusion"). One may also need to look at the concept of "work" as understood by the GPL (which or what "work" is the GPL talking about?), which is relevant for Art. 2 of the license. For example, a relevant "work" may be a compiled binary which incorporates a GPL'd library. But could a work also be considered to contain a GPL'd library merely because this library

23 E.g in Spain, under Articles 8 and 10 of the Spanish Copyright Law RDL 1/1996.

is loaded at run-time? Or the work may interact with GPL'd dependencies (libraries, whatever) that may be distributed separately (or downloaded separately) but are still required by the new work in order to function (disregarding operating system components,[24] though even that is a question that must also be answered).

An interesting and valuable view on the concept of "work" is contributed by the FSF itself, when commenting on "mere aggregation" in its GPL FAQs:[25]

> *What constitutes combining two parts into one program? This is a legal question, which ultimately judges will decide. We believe that a proper criterion depends both on the mechanism of communication (exec, pipes, rpc, function calls within a shared address space, etc.) and the semantics of the communication (what kinds of information are interchanged).*
>
> *If the modules are included in the same executable file, they are definitely combined in one program. If modules are designed to run linked together in a shared address space, that almost surely means combining them into one program.*
>
> *By contrast, pipes, sockets and command-line arguments are communication mechanisms normally used between two separate programs. So when they are used for communication, the modules normally are separate programs. But if the semantics of the communication are intimate enough, exchanging complex internal data structures, that too could be a basis to consider the two parts as combined into a larger program.*

So, we ask if the wording of GPLv2, in particular, means that the scope of its copyleft provisions apply to the combined work – whether statically, dynamically linked or otherwise related.

A particular issue with GPLv2 revolves around whether the courts of any jurisdiction would interpret its wording to extend to works which have a connection with a GPL'd work (interact), but are not derivative works *per se* as a matter of law, or their use otherwise does not require the GPL'd code author's consent under copyright law (compilations, collective works, etc.). Thus, possibly enforcing contractual control over the use of the work (which in fact runs against the stated purpose of the license: Article 0 clearly announces: "*Activities other than copying, distribution and modification are not covered by this License; they are outside its scope*"). The wording of GPLv2 is open to interpretation on this point.[26]

This issue is best explained by way of example:

24 GPLv2 Clause 3, second separate paragraph.

25 See <http://www.gnu.org/licenses/old-licenses/gpl-2.0-faq.html#MereAggregation>.

26 I believe that GPLv3 addresses this point more directly but still not necessarily in a clear manner. Clause 5: "*A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate*". Note the exclusion of "independent works" which ARE combined with the GPL'd work such as to form a larger program. See also the definition of Corresponding Source Code (including shared library files) in clause 1.

> *A coder takes a work subject to GPL2 ("X"), and incorporates a very small part of it ("P") into another work, ("D"). We have selected P such that incorporation of P in D does not, as a matter of copyright law, require the licence of the original copyright owner of X (this may be because P does not meet the threshold requirements in a particular jurisdiction to attract copyright protection – perhaps it lacks sufficient originality – or because of fair dealing or similar exemptions). It is uncontroversial, that, as a matter of copyright law, the exploitation of D does not require the consent of the copyright owner of X (and follows from our definition of "derivative work"). The question, however, is whether such exploitation of D is a breach of the licence under which X itself is exploited. This issue arises from wording in section 2(b) of GPLv2 which refers to a "work...that...contains...the Program or any part thereof".*

In other words, by distributing a non-derivative work D, may the coder still be in breach of GPLv2 as it applies to X? So, on a licence basis, could the coder potentially lose her licence to X, if this licence purports to require obligations which restrict otherwise-unrestricted acts, and she performs such an act? The answer to this may depend on whether the terms and conditions of the GPL are considered to be a licence or a contract – and if this has been validly formed, etc., but this may not be the only issue. If the answer to this question is "yes", then we have to look further than copyright law at the relationship between X and D to determine whether the GPL is breached in respect of X, if D is distributed other than under the GPL. If the answer is "no", then we only have to consider whether D is a derivative work or otherwise covered by X's copyright rights (in accordance with our definition). So, the question may not necessarily be only "what is one permitted to do by the license in terms of P or D (in our example)?", but also "how does exploitation of P/D affect the licence for X?". This subtle distinction should be taken into account when considering each form of software interaction.

## 2.2. Further comment

At each of these stages, the specific interaction at which we are looking could "fall by the wayside" in GPL v2 copyleft terms, as either copyright protection is not granted, or if it is, there is an exemption, or finally, the license itself provides for exemption from copyleft obligations.[27]

In addition, we must add that the GPL, as a copyright license, must be interpreted in each jurisdiction under the applicable laws in force (with the additional cross-border complication of determining which law should apply under conflict of law/private international law rules). The US and EU member state laws differ, particularly in their respective formal definitions of "*derivative works*", "*collective works*" and "*composite/composed works*".[28]

And another layer of complication is created if the GPL is considered a contractual document, to which varying jurisdiction-specific rules of contractual interpretation (*contra proferentem*, intention of the parties, etc.) may apply.[29]

27 For the sake of discussion: a wider interpretation of the license may rely on the use of functional or factual elements of expression of the GPL'd code, which may run against copyright principles which do not protect these parts, even if they are re-incorporated into the new work.

28 The Software Interactions Document provides a legal glossary at the end that discusses these terms.

29 See discussion of GPL as contract, e.g. Moglen, Eben (2001) *'Enforcing the GPL'*, online at <http://www.gnu.org/philosophy/enforcing-gpl.html>; Guadamuz, André (2004) *'Viral contracts or unenforceable*

## 3. Example: static linking

As an example of the analysis undertaken, this section presents the interaction mechanism commonly called "*static linking*".

In static linking, after the source code is compiled into object files, a linker will combine these object files into one executable at build time ("build time linking", as opposed to "load time linking"). Basically, the linker will copy into the executable the required instructions, data and other symbols of the linked file (and any further object files on which this linked file depends). This one executable will contain the machine code of all the components of the programs that were included in the *link* step.[30]

An executable is generally considered (by the legal community interested in FOSS) to be a *derivative work* of the programs and libraries contained in the executable – i.e. those that are statically linked into the executable, mainly because this is done by way of reproduction and transformation of those components.

Answering the 5 key questions:

1. The statically linked library is protected by copyright. As a whole, this would include its header information.

2. Copyright in the static library is indeed infringed in the static linking and redistribution of the library through reproduction and arguably modification of the library. The reproduction right is certainly relevant, and arguably the transformation right, it being argued that linking and compiling the library into the executable creates a derivative work of the library (see below). The distribution right is involved, as the library code would be redistributed as part of the executable.

3. Arguably we don't need to look at any further question, as this software interaction falls clearly within copyright as per answer to Question 1.[31]

4. By incorporating the whole library, no specific exemption may be available (though there have been arguments that even this form of combining is still only "using" the work as contemplated by the author, thus the interaction could be considered fair use (to be read in the light of the GPL, see next questions). Under a free software license, any user has permission to carry out these acts subject to compliance with applicable obligations in the event of distribution.

5. If the library has a copyleft license such as the GPL, the obligations as to redistribution depend on whether one considers that libraries are merely reproduced in the executable or are (also) transformed, as we discuss next.

*documents? Contractual validity of copyleft licences'* 26 European Intellectual Property Review 8 331.

30 This is called "*resolving the dependencies*", by automatic inclusion from external files or libraries in order to satisfy dependencies between the core program and the libraries.

31 See discussion below on wider application of copyright law.

As we have noted, it is generally thought that the process of static linking transforms the library, and thus creates a derivative work. Even if this is not the case, there are also arguments to say that the resulting executable "contains the library" (a collective work or compilation) and thus the executable is a "work based on the program" subject to copyleft obligations or Art. 2 GPLv2 on redistribution. In these circumstances, in order to be permitted to redistribute the GPL library, Art. 2 GPLv2 requires the whole work (the executable) to be redistributed under the GPL.

This is further reinforced by the expressed intent of the GPL "*to control the distribution of **derivative or collective** works based on the Program*".[32] As an executable with statically linked libraries contains code from those libraries, it is generally thought that this executable should, when distributed, be licensed under the GPL.

Legal appreciations vary.[33]

- Whether the original source code of the linking file (i.e. prior to link time) is a derivative work of a statically linked library can be questioned. All it does is reference external required code (symbols, header information of the library), it does not reproduce the linked code in any manner nor does it transform it.
- However, there is an argument that, as the program that contains this source file is designed and written to work with the external library code, it is then dependent on or "based on" the external library (i.e. it is not independent).
- This argument is in turn opposed by a counter argument that the linking program depends more on the interface specifications of the library and/or on the functionalities of the linked library which in both cases are arguably not protected by copyright laws (this may be a stronger argument in the US than in the EU, the US regime excluding "*procedures, processes, systems, methods of operation*" from copyright protection). In this view, creating symbols to refer and thus link to these functionalities is not an act restricted in any way by copyright or, if it were, it should be covered by the doctrine of *fair use*. In addition, it is arguable that – certainly within the EU – the symbols that are used to create the link are themselves protected, being interoperability (interface) information.
- A further argument holds that statically linking the code of a library to create an executable is the expected and normal "use" of a library and thus creating the executable does not entail transforming the library in any way – merely reproducing elements of the library in the executable. Thus the copyleft obligations of Art. 2 GPLv2 do not arise, despite the wording as to "containing the library" as there is no modification (a prior requisite for Art. 2), and rather the obligations under Art 1. (copying and distributing) will apply.

However, despite these arguments, in our experience most lawyers interested in this topic would tend to advise that the fact that the library is statically linked, (i.e code is added to the executable), results in a whole that is derivative of the library.

32 GPLv2, Art. 2.

33 E.g. see discussion in Rod Dixon (2003) *'Open source software law'*, Artech House, at p. 32 et seq. or Välimäki, Mikko (2005) ibid; Katz, Andrew (2007) ibid.

Finally, we should look at the scope of the concept “Work” in Art. 2: it could cover not just the new code, but the combined work of new code plus original GPL'd software artefact. It is this “whole” that must be distributed under terms compatible with the GPLv2, including all its component parts.

## 4. Separation and independence

As is set out in the main articles in the Software Interactions Document on the different interaction mechanisms, it is argued that there are some combinations of software programs that will generally always produce a derivative work, while other forms may not. But the dividing line between the two is not clear and in fact will depend on the facts of each case.

For example, one of the major arguments in this area has been that dynamic linking – which does not involve a transformation or compilation/linking of the linked code/library at development or build time - does not necessarily create a derivative work of that code/library, and the external library is only reproduced and distributed (Art. 1, GPLv2), rather than transformed and distributed (Art. 2, GPLv2).[34] Although it is then reproduced and linked at run-time (which might create a derivative work), this is only created in the user's computer memory, after redistribution.

If this is correct, the “strong copyleft” view, in order to apply conditions to the distribution of code dynamically linking to the library, may then have to rely on two arguments:

- “collectivity” (for lack of a better word): the dynamically linked library or plug-in is distributed along with the application code that uses it, as an integral part of the “combined” program, and the linking program is not an “independent and separate” work in itself. In this case, the GPL would apply to all the program that is distributed, not just the GPL'd library.
- “interdependency”: the main program that uses the GPL'd library is designed and written to include and use the functionalities of the external library (at runtime) and thus “depends” on the library to work. In this manner, an interdependent compilation has been created, which is argued to fall under the copyleft rules of the GPL.

Neither of these is necessarily a strong or definitive argument, as the new code could be written to a public API and use the GPL'd library as an implementation (among others) of that API. In addition, as we have already mentioned above in respect of statically linked libraries, writing code to use a library (and then executing the library at runtime) could be considered merely “using” the library in the intended manner covered by forms of “fair use”, as well as specifically excluded from the GPLv2 license conditions when it says: “the act of running the Program is not restricted” - thus requiring merely a consent to reproduce (but not modify) and redistribute the artefact. This argument has been set out in the main article of the Software Interactions Document on dynamic linking.

The “dependency” argument is of interest. It has been argued that if the new program is specifically designed and written to work (only) with certain libraries (or vice versa, it is designed

34 Discussed by Dixon Rid (2003), ibid at p32 et seq; Katz, Andrew (2007) ibid; Rosen, Lawrence (2001), ibid.

to be part of an existing third party program, (e.g. like a plug-in), and has little if no other use in any other context), then the program should indeed be considered "based on" (in a contractual meaning, if not a copyright meaning) the third party work. Against this argument, if, in the new work, one could substitute a third party library with another (older, newer, another operating system function, whatever), then it is more likely that the new work would be considered independent of the third party component (and thus either not derivative, or excluded by the "independent" wording of GPLv2).

So in all events the questions of separation, as regards functionalities, design and architecture, etc., and independence between programs / components both at design and development time are relevant questions and, while only based on hypothetical cases provided by our technical colleagues, the Software Interactions Document tries to look at them in each case.

## 5. Primary and Secondary infringement

Subject to the issues above relating to scope, the Software Interactions Document only considers primary infringement. In other words, potential infringement of copyright by reproduction, transformation (including translation, adaptation and arrangement) and distribution to the public[35].

In certain jurisdictions, secondary infringement of copyright is also unlawful (for example, in the United Kingdom, the Copyright, Designs and Patents Act 1988, section 16(2) provides that copyright is infringed by someone who "*without the licence of the copyright owner...authorises another to do...any of the acts restricted by copyright*". Other jurisdictions have similar provisions). Thanks, by and large, to litigation from rights owners of music and video content who are seeking to prevent the unauthorised distribution of their material by claiming secondary infringement against entities facilitating the unlawful distribution (but who do not themselves distribute – such as holders of peer-to-peer indices) the scope of secondary infringement at law is constantly changing.

It has been argued, for example, that distributing the Linux kernel together with an NDISWrapper amounts to secondary infringement.[36] The code of NDISWrapper is released under GPLv2. The argument runs along the lines that, where the Windows XP driver is not available under the GPL, the mere distribution of NDISWrapper somehow authorises a breach of GPLv2 as applicable to the kernel, in that it allows the Windows XP Driver to be interfaced (dynamically, as it happens) to kernel code, and that authorisation of that breach is, therefore, a secondary infringement.

The Software Interactions Document being concerned with primary infringement, no opinion is expressed as to the validity (or otherwise) of that argument. From our example, NDISWrapper itself, in the context of this document, and so far as primary infringement is concerned, can be analysed both from the perspective of a kernel module, and, to the extent that it (potentially) interfaces with non-GPL code, through its interaction with the Windows XP driver by dynamic

35 As set out in Article 4 of EUCPD.

36 An NDISWrapper in this case, is a driver which acts as an interface between the Linux kernel and Microsoft's Windows XP Driver Model interface, and enables hardware for which a Windows XP driver is available, but not a Linux driver, to work with Linux, by using the Windows XP driver instead of the native Windows driver. See debate referred to above, online at <http://kerneltrap.org/Linux/NDISwrapper_and_the_GPL>.

linking. It is argued that *shims* (pieces of code that are themselves typically released under the GPL, but act as an interface to non-GPL code, and of which the NDISWrapper is a specialised example) should be considered similarly.

## 6. Looking forward

As will be understood from this brief overview of the work that has been carried out on the Software Interactions Document, no definitive answer to the debate has been reached, though hopefully we have provided some useful pointers. Despite the hours of debate and significant number of missives discussing the issue and particular cases, I think we can still safely say that there is no black and white answer, though with luck we have been able to reduce some areas of grey. Uncertainty (from a legal point of view) is one of the major concerns of free software licensing, and we believe that anything that helps to reduce it will be beneficial for the community and industry in general. Having some pointers should help a project weigh up each case and make a more informed decision based on the merits.

As we mentioned above, one of the key objectives of the Document is educational, not just via the discussion of the software interactions covered in the paper, but also via the glossaries at its end, which should provide a common vocabulary on which engineers and lawyers may base their discussions.

As a next stage or step, we can think of two areas of work. First, it would be interesting to take several specific “real world” cases of software interactions (easily done, using publicly available code in any free project), to test the hypotheticals postulated here. Secondly, it would be interesting to expand the analysis, if found useful, with respect to GPLv3 and see what clarification is brought by the more modern wording of this license.

More practically, the Document is a work in progress, and we need more examples and/or diagrams that can help understand the technical issues involved (using header file data, published APIs, etc.) - something that might even be used as a model for presenting and arguing a case either between parties or before the courts.

From a purely legal point of view, bearing in mind the complication of dealing with over 25 jurisdictions (in Europe alone), we believe it would be useful to incorporate further work on the definitions of the legal concepts that are involved, in particular the concept of derivative and collective (composed) works in an IT context. In this respect, the IFOSS Law Book (another project promoted by the FSF-E[37]) is a useful starter and we look forward to taking advantage of synergies to improve the current work.

37 Online at http://ifosslawbook.org/

## About the author

***Malcolm Bain*** *is partner of Barcelona boutique IP/IT law firm "id law partners" and lecturer in the legal issues of FOSS at the Open University of Catalonia and the University of Lérida.*